AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions and listings of claims in the application.

Please cancel claims 3, 8, and 12 without prejudice.

1. (Currently amended) A computer-implemented method for optimizing an executable program having a plurality of functions and at least one function with a first name associated with executable code that implements the function at a first address and at least one linkage stub code segment having code that branches to the first address and a symbolic name by which the function is invoked in the program, comprising:

searching a symbol table for an entry having a symbolic name that is a derivation of the first name and reading a linkage stub address associated with the symbolic name;

identifying branch instructions having target addresses that reference the linkage stub code segment; and

replacing the target addresses of the branch instructions with the first address, wherein the first address is an address at which the code that implements the function is stored.

2. (Original) The method of claim 1, further comprising replacing the target address of the branch instructions with the first address only in functions that are reached during program execution.

3. (Canceled)

4. (Currently amended) The method of claim 1, ~~further comprising:~~

~~searching a symbol table for an entry having a symbolic name that matches the first name with~~ wherein the derivation includes an underscore prefix ~~and reading a linkage stub address associated with the symbolic name; and~~

~~replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored~~.

5. (Currently amended) The method of claim 1, ~~further comprising:~~

~~searching a symbol table for an entry having a symbolic name that matches the first name with~~ wherein the derivation includes an underscore suffix ~~and reading a linkage stub address associated with the symbolic name; and~~

~~replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored~~.

6. (Original) The method of claim 1, further comprising:

replacing function entry points in the executable program with breakpoints, whereby breakpointed functions are generated; and

upon encountering a breakpoint of a breakpointed function during program execution, identifying within the breakpointed function branch instructions that target linkage stub functions.

7. (Original) The method of claim 6, further comprising:

storing original instructions from the function entry points prior to replacement with the breakpoints;

upon encountering a breakpoint of a breakpointed function during program execution, restoring the original instruction to the entry point of the breakpointed function.

8. (Canceled)

9. (Currently amended) The method of claim 6, ~~further comprising:~~

~~searching a symbol table for an entry having a symbolic name that matches the first name with~~ wherein the derivation includes an underscore prefix ~~and reading a linkage stub address associated with the symbolic name; and~~

~~replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored~~.

10. (Currently amended) The method of claim 6, ~~further comprising:~~

~~searching a symbol table for an entry having a symbolic name that matches the first name with~~ wherein the derivation includes an underscore suffix ~~and reading a linkage stub address associated with the symbolic name; and~~

~~replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored~~.

11. (Original) The method of claim 1, further comprising:

replacing entry points of linkage stub code segments in the executable program with breakpoints, whereby breakpointed linkage stubs are generated; and

upon encountering a breakpoint of a breakpointed linkage stub during program execution, changing a target address of a branch instruction that branched to the breakpointed linkage stub to reference the function referenced by the breakpointed linkage stub.

12. (Canceled)

13. (Currently amended) The method of claim 11, ~~further comprising:~~

~~searching a symbol table for an entry having a symbolic name that matches the first name with~~ wherein the derivation includes an underscore prefix ~~and reading a linkage stub address associated with the symbolic name; and~~

~~replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored~~.

14. (Currently amended) The method of claim 11, ~~further comprising:~~

~~searching a symbol table for an entry having a symbolic name that matches the first name with~~ wherein the derivation includes an underscore suffix ~~and reading a linkage stub address associated with the symbolic name; and~~

~~replacing target addresses of branch instructions having target addresses equal to the linkage stub address with an address at which the code that implements the function is stored~~.

15. (Currently amended) An apparatus for optimizing an executable program having a plurality of functions and at least one function with a first name associated with executable code that implements the function at a first address and at least one linkage stub code segment having code that branches to the first address and a symbolic name by which the function is invoked in the program, comprising:

means for searching a symbol table for an entry having a symbolic name that is a derivation of the first name and reading a linkage stub address associated with the symbolic name;

means for identifying branch instructions having target addresses that reference the linkage stub code segment; and

means for replacing the target addresses of the branch instructions with the first address, wherein the first address is an address at which the code that implements the function is stored.

16. (New) A computer-implemented method for optimizing an executable program, comprising:

generating the executable program code from a program that includes calls to functions in a library that is external to the program;

identifying linkage stub segments in the executable program, wherein each linkage stub segment is limited to transferring control to executable code that implements a respective function of the library at a respective function address;

identifying in the executable program, branch instructions having target addresses that reference the linkage stub segments;

replacing in each of the identified branch instructions, the respective target address that references one of the linkage stub segments with the respective function address of the executable code that implements the respective function.

17. (New) The method of claim 16, wherein identifying a linkage stub segment includes searching a symbol table for first name that is a derivation of a second name of the respective function at the respective function address, and reading a linkage stub address associated with the first name.

18. (New) The method of claim 17, wherein the derivation includes one of an underscore prefix and an underscore suffix.

19. (New) The method of claim 16, further comprising:

replacing function entry points in the executable program with breakpoints, whereby breakpointed functions are generated; and

upon encountering a breakpoint of a breakpointed function during program execution, identifying within the breakpointed function branch instructions that target linkage stub functions.

20. (New) The method of claim 19, further comprising:

storing original instructions from the function entry points prior to replacement with the breakpoints;

upon encountering a breakpoint of a breakpointed function during program execution, restoring the original instruction to the entry point of the breakpointed function.